Optimal Unknown Bit Filtering for Test Response Masking

Ding-ke Weng and Jiann-Chyi Rau
Department of Electrical Engineering Tamkang University
Tamsui, Taipei, Taiwan
{dingklewong@hotmail.com , jcrau@ee.tku.edu.tw}

Cheng-han Lin
Department of Electrical Engineering Tamkang University
Tamsui, Taipei, Taiwan
milk0919520173@hotmail.com

Abstract—**In this paper presents a new X-Masking scheme for response compaction. It filters all X states from test response that can no unknown value input to response compactor. In the experimental results, this scheme increased less control data and maintain same observability.**

Keywords—response unknown; x-masking; compactor; response compaction

1. INTRODUCTION

As designs grow in size, it becomes increasingly expensive to maintain a high level of test coverage. This is due to a prohibitively large volume of test data and long test application times. Accordingly, the methods employed to reduce the amount of test data are instrumental in maintaining the high efficiency of testing schemes. Test response compaction, in conjunction with stimuli compression, plays a crucial role in handling test data volume growth. Unfortunately, in many designs, unknown (X) states can be injected into a compactor, where they severely affect a signature. This applies primarily to *time compactors* which allow X states to quickly multiply (due to a feedback fan-out) and sustain their presence until a read out operation. Interestingly, we can add some simple circuit or modify time compactor like *modular time compactors* [1] prevent multiplication of X states.

Space compactor is different with *time compactor*. It can tolerate a certain amount of X states. Actually, *Space compactor* is combinational compactors, it reduces a negative impact of X states to a single scan shift cycle [2], [3]. To avoid masking, however, they must observe each scan chain on two or more outputs. But even state-of-the-art compactors do not tolerate Xs to a high enough standard for today's needs.

Many schemes addressing selective observation of scan chains (scan cells) have been proposed to date. For instance, an OPMISR [7] employs a circuitry to mask selected unload values so that X states do not reach a compactor. Scan chains partitioned into a minimal number of fragments employ LFSR reseeding driving masking in [10]. The X-block method [11] uses the same technique but requires less control data and simpler test logic. The unknown blocking scheme of [12] deploys the LFSR reseeding as well. [13] the masking signals are formed by AND-ing several outputs of a phase shifter to decrease probability of blocking non-X responses.

In above technique the circuit used AND logic or OR logic that introduced between the circuit under test (CUT) and the compactor is X-masking logic (XML). Any logic BIST or test data compression scheme may provide inputs to an XML proposed in [15]. A somewhat similar test set-dependent masking circuitry is described in [16]. A channel masking scheme of [17] allows either disabling all scan chains, or selecting scan chains belonging to one of two groups at the price of possible over-masking. If miss detect faults, regenerate decision pattern recover fault coverage. An X-tolerant MISR used in [18] rests on a weighted random pattern generator with no data transfer between its memory elements.

In this paper, we present a new scheme is a novel algorithm. It can handle any percentage of X state. It can easily reduce date volume cause novel architecture. Major original contributions of the paper include:

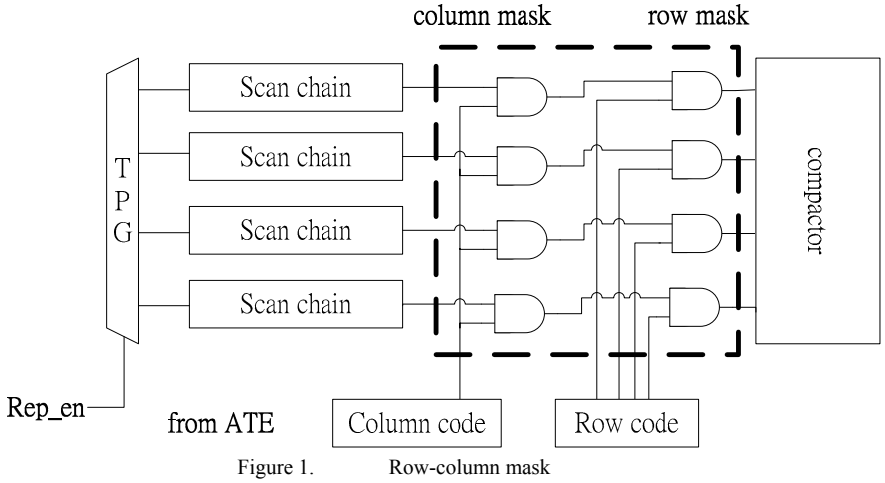

Figure 1. Row-column mask

1. How to mask X states in the novel architecture?
2. How to generate mask control code in the novel architecture?
3. How to preserve original fault coverage by generates additional stimulus pattern?

II. ROW-COLUMN XML ARCHITECTURE

Row-column XML architecture different to traditional masking logic, it separates into two part, column mask (horizontal) and row (vertical) mask.

Column mask input control code one bit when shift cycle, code numbers depend on how many depth with the scan chain. If one bit of column code set 0, then one slice of scan chain will be masked.

Row mask change control code at capture cycle maintain until scan chain shift out, code numbers depend on the number of scan chain. If row code set 0, one scan chain will all be masked.

In this architecture, while the flip-flop captures unknown value, the row or column must one mask be masking.

III. GENERATE MASK CODE

A flip-flop can detect many faults, this because a faulty effect activated at circuit often propagated through several paths. It's like Fig. 2, fault effects captured in scan flip-flops by test pattern. Totally can detect eight faults f1, f2, … …, f8 if can directly observe without any compactor. Our goal is producing a mask code can observe eight faults.

For convenient to calculation, assuming can detect a fault is setting value 1. The cell value of matrix is number of detecting faults in every flip flop. If can't detect faults, it means is unknown value, therefore the value is 0.

The Fig. 3 is transferred from Fig. 2. If the flip-flop coordinates expressed as (chain, slice), this pattern unknown node is (1,2), (1,3), (2,3), (3,3), (3,4). Others cell at least can detect one fault.

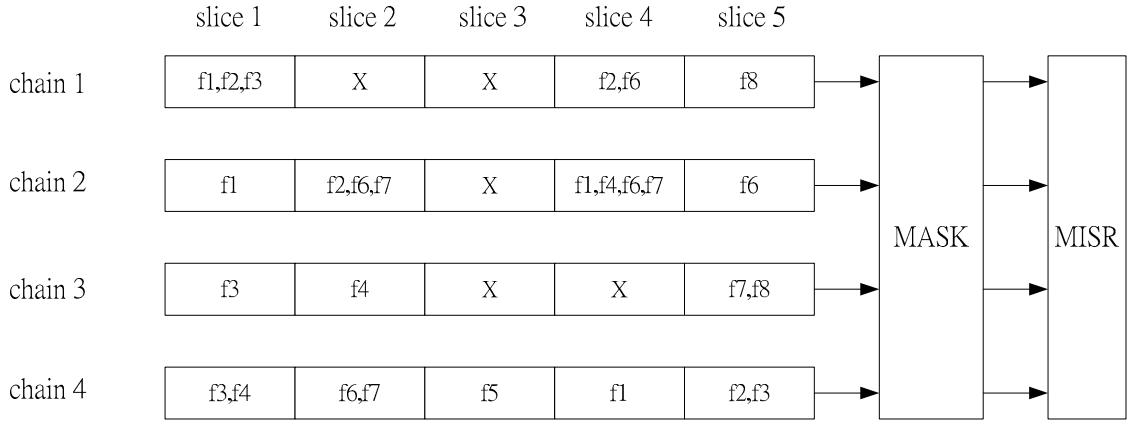

Figure 2. Every flip-flop detect fault

	s1	s2	s3	s4	s5
c1	3	0	0	2	1
c2	1	3	0	4	1
c3	1	1	0	0	2
c4	2	2	1	1	2

Figure 3. Matrix of value

If we select mask control is setting scan chain 1, 2, 3 be masked, the result is like Fig. 3, gray color means be masked cell. All the X-response can be masked, but as same happen in non-X data. It maybe losses fault coverage.

One pattern not only one mask control, may have many possible mask code. We need select one can obtain maximum total value.

Assuming scan chain has m rows, and n columns. r_i and c_j are row code and column code. When searching control mask code, we must follow following rules.

if $w_{ij} = 0$ then $r_i \bullet c_j = 0$

$$Maximum: \sum_{i=1}^{i=m} \sum_{j=1}^{j=n} w_{ij} r_i c_j \qquad (1)$$

A. Greedy algorithm

Either the column or row that X in must be masked. To choose which one shall be masked the sum of column or row should be calculated. The less one will be chosen to be masked and another will be kept. But we do not consider how many X there are in this method. Sometimes X exists in the same column, if we mask this column all X blocked but we may mask many rows instead.

Opposite, consider X number for select mask code, can less line be masked, but not purpose to obtain maximum total value. And quantity of X in column and in row is easily the same.

B. Expectation algorithm

1) Formula origin

Combine above experiences of produces mask control code. Create a formula similar to probability of expectation. We called the line that is a column or row for convenient to explain formula. The formula calculates a line expectation. Expect obtains how many value if the line has been masked.

If a cell be masked due the line that expectation is minus the cell's value. Opposite, the line vertical to the line that has been masked due X already be masked so can be

maintained. The line expectation adds sum of the vertical line.

Sometimes the vertical line can be maintained conditional on more than one X. It may need two or more line be masked. Others line probability to be masked, we assumed that the probability conditioned by X number on the line want be maintained.

This function replaces using algebraic formulas is shown by (2).

$$E_n = -W_{sum} + \sum_{every\perp line}^{nth==X} \frac{1}{Xnum_{\perp}}(W_{sum\perp} + \sum_{every\perp line}^{mth==X} E_{//}) \quad (2)$$

W_{sum} is sum of value on the line

X_{num} is number of X on the line

$\perp$ is the line is vertical to the line that we want to calculate expectation.

// is the line want to be masked in order to keep vertical line not be masked.

The '$\sum$' is every the line that is vertical line nth number is X state on the line. The nth number depend on what line is calculating in front of the $\sum$.

$E_{//}$ is the line that parallel to want to calculates line its expectation.

In the formula, if the line has been calculated, it becomes 0. It is in order to avoid the formula for infinite repeat.

2) Execution procedure

The execution procedure of computing control code:

Step 1: computing every line expectation, the line exist x state.

Step 2: choose the line is maximum expectation. Set mask control code, and change matrix value.

Step 3: repeat steps 1 and 2, until the matrix without 0.

Step 4: finish code procedure.

Step 1 is searching all lines exist 0 and then calculate the line expectation, not exist 0 lines shouldn't mask so without calculating. Start step 2 when get all line expectation. Choose the line is maximum expectation to mask. Set that line control code to 0. Change point 0 to -1, and others masked point change to minus value (actual minus value can decided by yourself. It will get different answers, but don't know the best value).

Using has been changed matrix repeat step 1 and 2 until matrix without 0 points, and then, this pattern mask code is finished.

3) Example

For example with Fig.3 pattern that have six lined r1, r2, r3, c2, c3, c4 need to calculate. Every row expectation shows at table left side and column expectation shows at table top side.

At first calculating, the maximum expectation is s3, through masked the matrix change to Fig. 4(a). In the rest unknown through calculation choice c3 to mask, show Fig.4 (b). At last select masked s2, results shown in Fig.4 (c). And the row code is 1101, column code is 10011.

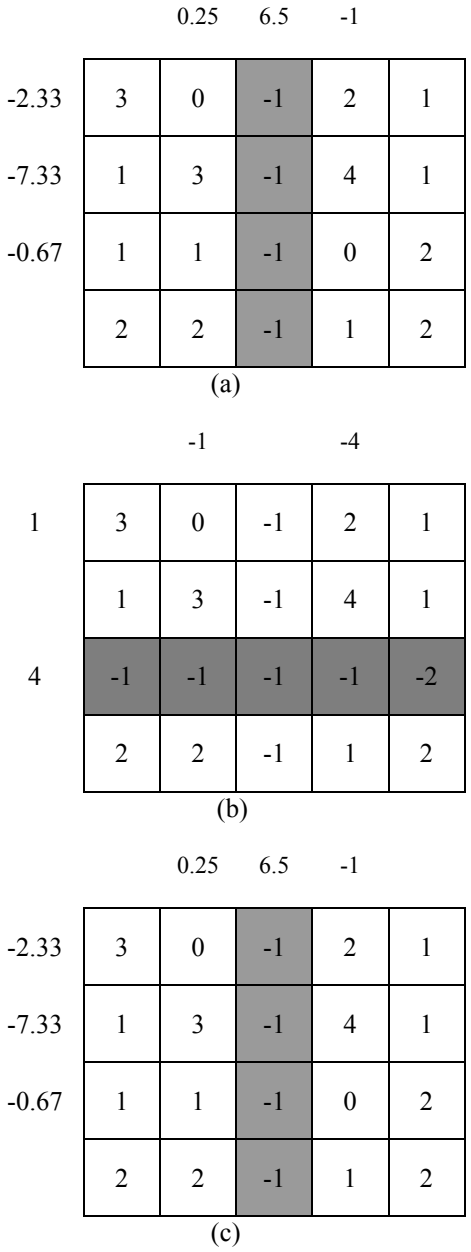

Figure 4. Matrix of changes when calculating mask code

4) Simplify

The expectation formula in a lot of X case, the formula will be expanded very huge. The calculation becomes very time-consuming, because the formula is completed by repetitional iteration.

Every time expectation divided by $X_{num\perp}$, the back value effecting is limited on the result. We assume iteration nth time meaning the formula is n order. The higher order effective is less than lower.

By the way, the formula calculates the answer will be different in different order. We think expectation consider extensive area in higher order, opposite decision directly in low order.

5) Optimization

In additional, after experiment found scan chain more close to square that mask code data volume less. Because

the largest factor is its answer's variation is less. If row length and column length gap is very huge, mask code would favor to mask a shorter part. So row and column length are similar can observe more detect fault.

The modify method is adding extra flip-flops for constructing a closer to square matrix. For example, the original scan chain is 33 flip-flops, the matrix is 3X11. But adding 2 flip-flops become to 35, the matrix is 5X7. Every pattern its mask code less 2 bits.

IV. FAULT COVERAGE

Typically, [12] a set of test patterns that is generated by an ATPG to achieve high fault coverage detects most faults multiple times. In other words, most faults are detected by more than one test pattern in the set, this kind fault call *multiple detection fault*, otherwise, a fault detect in only one test pattern in the set is call *single detection fault*.

A. Recover fault coverage

If finally find lost some detection faults, we need generate additional pattern to detect. The directly solved method is repeat test pattern that can detect the fault. But calculating matrix still in the same values, the answer will be same. So need to modify value that want to detect point, it adds a large enough weighted value G. It's obtain by the rule of thumb or experimental. In principle, G need follow:

$$G \geq W_{sum} \tag{3}$$

W_{sum} denoted matrix total value of all points

3	0	0	2	1
1	3	0	4	1
1	1	0	0	2
2	2	51	1	2

Figure 5. A result of mask calculation after matrix added value G, gray part is behalf of be masked

B. Specifying Control Patterns

To identify the patterns that detect each fault, we conduct n-detection fault simulation and form fault list F that are detected by each test pattern when scan chains are directly observed without any compression scheme after the outputs of scan chains, the method from [11].

Fig. 6 assume the test set have 4 patterns, total detect f1, f2, f3,……., fn. At right hand shows allocation of all the fault type, the slash box denotes amount of single detection fault, the white box denotes amount of multiple detection fault. In p2, fc detect number is 2 but through p2 mask control fc be masked, it detect number minus 1 become single detection fault from multiple detection fault.

Sometimes even we add weighted value G, but mask control still masks the fault so the last fault list line shows still missed fault, need repeat above test pattern to keep the same fault coverage.

V. EXPERIMENTAL RESULTS

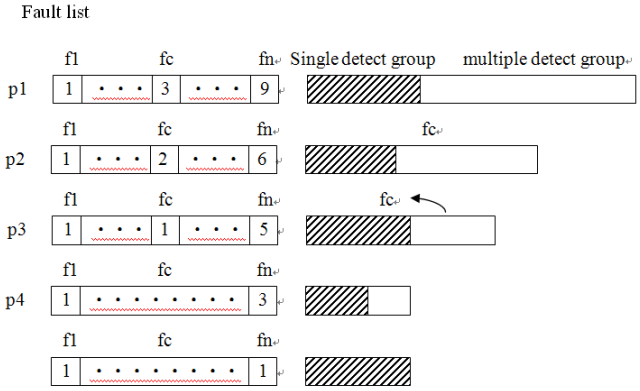

Figure 6. n-detection target fault list

Experimental results obtained by applying the proposed method to test patterns of ISCAS 89 benchmark circuits. The deterministic test patterns used in the experiments were generated by a commercial ATPG tool. The column #pattern gives the number of test patterns. The column #slice shows the slice numbers of scan chain. The column #chain shows the chain numbers of scan chain. The column Control data is total control data volume. The #repeat shows the number of repeat test patterns for keep the same fault coverage.

Table I shows used in exhaustive algorithm with section IV method retain fault coverage. The exhaustive algorithm just considers every mask probably for obtaining a max values sum. The table last column % denotes percentage of amount of control data relative to quantity of response bit. The larger circuit because too time-consuming so no experimental results.

TABLE I. exhaustive algorithm

circuit	#pattern	#slice	#chain	X%	#repeat	Control data	%
s1196	155	8	5	22.73%	7	2106	33.97%
s1238	161	7	6	22.26%	6	2171	32.11%
s1488	137	10	6	21.07%	22	2544	30.95%
s1494	131	10	6	20.95%	21	2432	30.94%
s5378	264	21	14	8.85%	55	11165	14.38%

Table II shows expectation algorithm experimental result in all circuits. We find the answer is close to exhaustive algorithm, some circuits better than exhaustive algorithm.

Table III is used focus specific fault control pattern method with expectation algorithm. The method can effectively detect all faults. That without the pattern that can't detect single detection fault, so decrease test pattern. The table last column % is control data bit relative to Table II control data bit.

Table IV is our proposed method compare with X-block method. We compare with the most high X percent experimental result that proposed in [11]. His test pattern X less, occur over-masking will less, so test pattern can less than me. But our proposed method indeed detects all faults. And our proposed method is advantage of speed. Because

of different technology, our method mask control code not need decompression, so do not need to spend time.

TABLE II. expectation algorithm

Table circuit	#pattern	#slice	#chain	X%	#repeat	Control data	FC%
s1196	155	8	5	22.73%	6	2093	99.9
s1238	161	7	6	22.26%	11	2236	99.9
s1488	137	10	6	21.07%	15	2432	99.9
s1494	131	10	6	20.95%	19	2400	99.9
s5378	264	21	14	8.85%	60	11340	99.9
s9234	395	21	18	10.94%	85	18720	99.9
s13207	475	40	26	12.17%	189	43824	99.9
s15850	469	33	25	10.53%	165	36772	99.9
s35932	69	64	32	10.56%	237	29376	99.9
s38417	983	46	41	17.20%	762	151815	99.9
s38584	708	67	36	10.28%	394	113506	99.9

TABLE III. expectation algorithm with specifying control patterns

circuit	#pattern	#slice	#chain	X%	#repeat	Control data	%
s1196	76	8	5	22.73%	5	1053	-49.69%
s1238	77	7	6	22.26%	0	1001	-55.23%
s1488	91	10	6	21.07%	9	1600	-34.21%
s1494	79	10	6	20.95%	15	1504	-37.33%
s5378	197	21	14	8.85%	28	7875	-30.56%
s9234	246	21	18	10.94%	55	11739	-37.29%
s13207	297	40	26	12.17%	156	29898	-31.78%
s15850	306	33	25	10.53%	117	24534	-33.28%
s35932	59	64	32	10.56%	789	81408	177.12%
s38417	705	46	41	17.20%	859	136068	-10.37%
s38584	626	67	36	10.28%	645	130913	15.34%

TABLE IV. compare with method [11]

	Proposed					[11] in 5% X			
name	#pat	X%	store bit	Time sec	FC%	#pat	store bit	Time sec	FC%
s5378	225	8.85%	8k	0	99.9	136	3672	4.3	99.0
s9234	301	10.94%	12k	0	99.9	210	6090	11.3	94.3
s13207	453	12.17%	30k	0	99.9	204	12648	19.1	98.4
s15850	423	10.53%	25k	0	99.9	161	9016	15.6	97.3
s38417	1564	17.20%	136k	0	99.5	189	22680	104	99.5
s38584	1102	10.28%	113k	0	99.9	195	25740	100	96.5

[1] W. Rajski and J. Rajski, "Modular compactor of test responses," Proc. VTS, pp. 242-251, 2006.

[2] S. Mitra and K. S. Kim, "X-Compact: an efficient response compaction technique," IEEE Trans. CAD, vol. 23, pp. 421- 432, March 2004.

[3] J. H. Patel, S. S. Lumetta, and S. M. Reddy, "Application of Saluja-Karpovsky compactors to test responses with many unknowns," Proc. VTS, pp. 107-112, 2003.

[4] T. Clouqueur, H. Fujiwara, K. Zarrineh, and K. Saluja, "Design and analysis of multiple-weight linear compactors of responses containing unknown values," Proc. ITC, pp. 1099-1108, 2005.

[5] J. Rajski, J. Tyszer, C. Wang, and S. Reddy, "Convolutional compaction of test responses," Proc. ITC, pp.745-754, 2003.

[6] C. Wang, et al., "On compacting test responses data containing unknown values," Proc. ICCAD, pp. 855-862, 2003.

[7] C. Barnhart, et al., "OPMISR: The foundation for compressed ATPG vectors," Proc. ITC, pp. 748-757, 2001.

[8] J. Rajski, et al., "Embedded deterministic test for low cost manufacturing test," Proc. ITC, pp. 301-310, 2002.

[9] H. Tang, et al., "On efficient X-handling using a selective compaction scheme to achieve high test response compaction ratios", Proc. VLSI Design, pp. 59-64, 2005.

[10] M. Naruse, I. Pomeranz, S.M. Reddy, and S. Kundu, "Onchip compression of output responses with unknown values using LFSR reseeding," Proc. ITC, pp. 1060-1068, 2003.

[11] S. Wang, K.J. Balakrishnan, and W. Wei, "X-Block: An efficient LFSR reseeding-based method to block unknowns for temporal compactors," IEEE Trans. Comput, vol. 57, pp. 978-989, July 2008.

[12] S. Wang, W. Wei, and S.T. Chakradhar, "Unknown blocking scheme for low control data volume and high observability," Proc. DATE, pp. 1 – 6, 2007.

[13] E.H. Volkerink and S. Mitra, "Response compaction with any number of unknowns using a new LFSR architecture," Proc.DAC, pp. 117-122, 2005.

[14] P. Wohl, J.A. Waicukauski, S. Patel, and A. Amin, "X-tolerant compression and application of scan-ATPG patterns in a BIST architecture," Proc. ITC, pp. 727-736, 2003.

[15] Y. Tang, et al., "X-masking during logic BIST and its impact on defect coverage," Proc. ITC, pp. 442-451, 2003.

[16] I. Pomeranz, S. Kundu, and S. M. Reddy, "On output response compression in the presence of unknown output values,"Proc. DAC, pp. 255-258, 2002.

[17] V. Chickermane, B. Foutz, and B. Keller, "Channel masking synthesis for efficient on-chip test compression," Proc. ITC,pp. 452-461, 2004.

[18] S. Mitra, S.S. Lumetta, M. Mitzenmacher, "X-tolerant signature analysis," Proc. ITC, pp. 432-441, 2004.

[19] Sinanoglu, O.; Almukhaizim, S, "X-Align: Improving the Scan Cell Observability of Response Compactors," IEEE Trans. Comput, pp.1392 – 1404, 2009